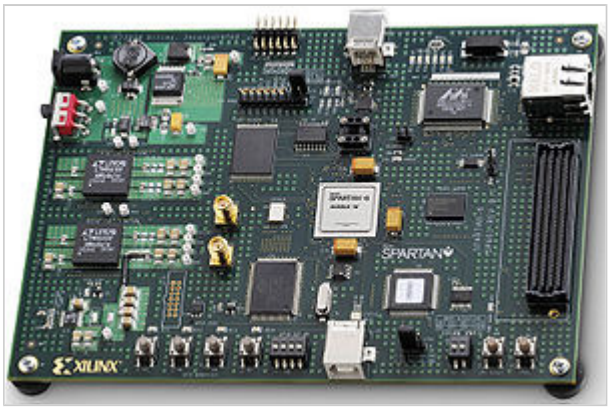
Field-programmable gate array

From Wikipedia, the free encyclopedia
(Redirected from FPGA)

An Altera Stratix IV GX FPGA

A **Field-programmable Gate Array** (**FPGA**) is an integrated circuit designed to be configured by the customer or designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the design[1] and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications.[2]



An example of an FPGA programming/evaluation board

FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"—somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations . Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.[2]

In addition to digital functions, some FPGAs have analog features. The most common analog feature is programmable slew rate and drive strength on each output pin, allowing the engineer to set slow rates on lightly loaded pins that would otherwise ring unacceptably, and to set stronger, faster rates on heavily loaded pins on high-speed channels that would otherwise run too slow.[3][4] Another relatively common analog feature is differential comparators on input pins designed to be connected to differential signaling channels. A few "mixed signal FPGAs" have integrated peripheral Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) with analog signal conditioning blocks allowing them to operate as a system-on-a-chip.[5] Such devices blur the line between an FPGA, which carries digital ones and zeros on its internal programmable interconnect fabric, and field-programmable analog array (FPAA), which carries analog values on its internal programmable interconnect fabric.

Contents

History

The FPGA industry sprouted from programmable read-only memory (PROM) and programmable logic devices (PLDs). PROMs and PLDs both had the option of being programmed in batches in a factory or in the field (field programmable), however programmable logic was hard-wired between logic gates.[6]

In the late 1980s the Naval Surface Warfare Department funded an experiment proposed by Steve Casselman to develop a computer that would implement 600,000 reprogrammable gates. Casselman was successful and a patent related to the system was issued in 1992.[6]

Some of the industry's foundational concepts and technologies for programmable logic arrays, gates, and logic blocks are founded in patents awarded to David W. Page and LuVerne R. Peterson in 1985.[7][8]

Xilinx Co-Founders, Ross Freeman and Bernard Vonderschmitt, invented the first commercially viable field programmable gate array in 1985 – the XC2064.[9] The XC2064 had programmable gates and programmable interconnects between gates, the beginnings of a new technology and market.[10] The XC2064 boasted a mere 64 configurable logic blocks (CLBs), with two 3-input lookup tables (LUTs).[11] More than 20 years later, Freeman was entered into the National Inventors Hall of Fame for his invention.[12]

Xilinx continued unchallenged and quickly growing from 1985 to the mid-1990s, when competitors sprouted up, eroding significant market-share. By 1993, Actel was serving about 18 percent of the market.[10]

The 1990s were an explosive period of time for FPGAs, both in sophistication and the volume of production. In the early 1990s, FPGAs were primarily used in telecommunications and networking. By the end of the decade, FPGAs found their way into consumer, automotive, and industrial applications.[13]

FPGAs got a glimpse of fame in 1997, when Adrian Thompson, a researcher working at the University of Sussex, merged genetic algorithm technology and FPGAs to create a sound recognition device. Thomson's algorithm configured an array of 10 x 10 cells in a Xilinx FPGA chip to discriminate between two tones, utilising analogue features of the digital chip. The application of genetic algorithms to the configuration of devices like FPGA's is now referred to as Evolvable hardware[6][14]

Modern developments

A recent trend has been to take the coarse-grained architectural approach a step further by combining the logic blocks and interconnects of traditional FPGAs with embedded microprocessors and related peripherals to form a complete "system on a programmable chip". This work mirrors the architecture by Ron Perlof and Hana Potash of Burroughs Advanced Systems Group which combined a reconfigurable CPU architecture on a single chip called the SB24. That work was done in 1982. Examples of such hybrid technologies can be found in the Xilinx Virtex-II PRO and Virtex-4 devices, which include one or more PowerPC processors embedded within the FPGA's logic fabric. The Atmel FPSLIC is another such device, which uses an AVR processor in combination with Atmel's programmable logic architecture. The Actel SmartFusion devices incorporate an ARM architecture Cortex-M3 hard processor core (with up to 512kB of flash and 64kB of RAM) and analog peripherals such as a multi-channel ADC and DACs to their flash-based FPGA fabric.

An alternate approach to using hard-macro processors is to make use of soft processor cores that are implemented within the FPGA logic.

As previously mentioned, many modern FPGAs have the ability to be reprogrammed at "run time," and this is leading to the idea of reconfigurable computing or reconfigurable systems — CPUs that reconfigure themselves to suit the task at hand. The Mitrion Virtual Processor from Mitrionics is an example of a reconfigurable soft processor, implemented on FPGAs. However, it does not support dynamic reconfiguration at runtime, but instead adapts itself to a specific program.

Additionally, new, non-FPGA architectures are beginning to emerge. Software-configurable microprocessors such as the Stretch S5000 adopt a hybrid approach by providing an array of processor cores and FPGA-like programmable cores on the same chip.

Gates

- 1987: 9,000 gates, Xilinx[10]
- 1992: 600,000, Naval Surface Warfare Department[6]
- Early 2000s: Millions [13]

Market size

- 1985: First commercial FPGA technology invented by Xilinx[10]
- 1987: $14 million[10]
- ~1993: >$385 million[10]
- 2005: $1.9 billion[15]
- 2010 estimates: $2.75 billion [15]

FPGA design starts

- 10,000[16]
- 2005: 80,000[17]
- 2008: 90,000[18]

FPGA comparisons

Historically, FPGAs have been slower, less energy efficient and generally achieved less functionality than their fixed ASIC counterparts. A study has shown that designs implemented on FPGAs need on average 18 times as much area, draw 7 times as much dynamic power, and are 3 times slower than the corresponding ASIC implementations.[19]

Advantages include the ability to re-program in the field to fix bugs, and may include a shorter time to market and lower non-recurring engineering costs.[*citation needed*] Vendors can also take a middle road by developing their hardware on ordinary FPGAs, but manufacture their final version so it can no longer be modified after the design has been committed.

Xilinx claims that several market and technology dynamics are changing the ASIC/FPGA paradigm:[20]

- Integrated circuit costs are rising aggressively
- ASIC complexity has lengthened development time
- R&D resources and headcount are decreasing
- Revenue losses for slow time-to-market are increasing
- Financial constraints in a poor economy are driving low-cost technologies

These trends make FPGAs a better alternative than ASICs for a larger number of higher-volume applications than they have been historically used for, to which the company attributes the growing number of FPGA design starts (see History).[20]

Some FPGAs have the capability of partial re-configuration that lets one portion of the device be re-programmed while other portions continue running.

Versus complex programmable logic devices

An Altera Cyclone II FPGA, on an Altera teraSIC DE1 Prototyping board.

The primary differences between CPLDs (Complex Programmable Logic Devices) and FPGAs are architectural. A CPLD has a somewhat restrictive structure consisting of one or more programmable sum-of-products logic arrays feeding a relatively small number of clocked registers. The result of this is less flexibility, with the advantage of more predictable timing delays and a higher logic-to-interconnect ratio. The FPGA architectures, on the other hand, are dominated by interconnect. This makes them far more flexible (in terms of the range of designs that are practical for implementation within them) but also far more complex to design for.

Another notable difference between CPLDs and FPGAs is the presence in most FPGAs of higher-level embedded functions (such as adders and multipliers) and embedded memories, as well as to have logic blocks implement decoders or mathematical functions.

Security considerations

With respect to security, FPGAs have both advantages and disadvantages as compared to ASICs or secure microprocessors. FPGAs' flexibility makes malicious modifications during fabrication a lower risk.[21] For many FPGAs, the loaded design is exposed while it is loaded (typically on every power-on). To address this issue, some FPGAs support bitstream encryption.[22][23]

Applications

Applications of FPGAs include digital signal processing, software-defined radio, aerospace and defense systems, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas.

FPGAs originally began as competitors to CPLDs and competed in a similar space, that of glue logic for PCBs. As their size, capabilities, and speed increased, they began to take over larger and larger functions to the state where some are now marketed as full systems on chips (SoC). Particularly with the introduction of dedicated multipliers into FPGA architectures in the late 1990s, applications which had traditionally been the sole reserve of DSPs began to incorporate FPGAs instead.[24][25]

FPGAs especially find applications in any area or algorithm that can make use of the massive parallelism offered by their architecture. One such area is code breaking, in particular brute-force attack, of cryptographic algorithms.

FPGAs are increasingly used in conventional high performance computing applications where computational kernels such as FFT or Convolution are performed on the FPGA instead of a microprocessor.

The inherent parallelism of the logic resources on an FPGA allows for considerable computational throughput even at a low MHz clock rates. The flexibility of the FPGA allows for even higher performance by trading off precision and range in the number format for an increased number of parallel arithmetic units. This has driven a new type of processing called reconfigurable computing, where time intensive tasks are offloaded from software to FPGAs.

The adoption of FPGAs in high performance computing is currently limited by the complexity of FPGA design compared to conventional software and the turn-around times of current design tools.

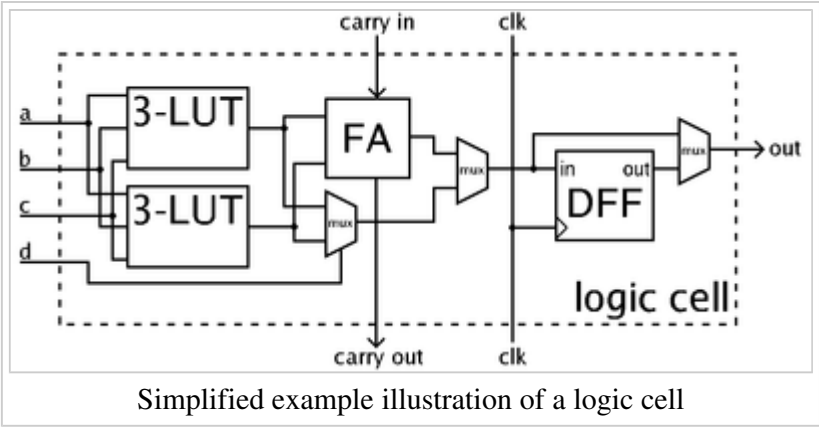
Traditionally, FPGAs have been reserved for specific vertical applications where the volume of production is small. For these low-volume applications, the premium that companies pay in hardware costs per unit for a programmable chip is more affordable than the development resources spent on creating an ASIC for a low-volume application. Today, new cost and performance dynamics have broadened the range of viable applications.

Architecture

The most common FPGA architecture[26] consists of an array of logic blocks (called Configurable Logic Block, CLB, or Logic Array Block, LAB, depending on vendor), I/O pads, and routing channels. Generally, all the routing channels have the same width (number of wires). Multiple I/O pads may fit into the height of one row or the width of one column in the array.

An application circuit must be mapped into an FPGA with adequate resources. While the number of CLBs/LABs and I/Os required is easily determined from the design, the number of routing tracks needed may vary considerably even among designs with the same amount of logic. For example, a crossbar switch requires much more routing than a systolic array with the same gate count. Since unused routing tracks increase the cost (and decrease the performance) of the part without providing any benefit, FPGA manufacturers try to provide just enough tracks so that most designs that will fit in terms of LUTs and IOs can be routed. This is determined by estimates such as those derived from Rent's rule or by experiments with existing designs.

In general, a logic block (CLB or LAB) consists of a few logical cells (called ALM, LE, Slice etc). A typical cell consists of a 4-input Lookup table (LUT), a Full adder (FA) and a D-type flip-flop, as shown below. The LUTs are in this figure split into two 3-input LUTs. In *normal mode* those are combined into a 4-input LUT through the left mux. In *arithmetic* mode, their outputs are fed to the FA. The selection of mode is programmed into the middle mux. The output can be either synchronous or asynchronous, depending on the programming of the mux to the right, in the figure example. In practice, entire or parts of the FA are put as functions into the LUTs in order to save space.[27][28][29]



Simplified example illustration of a logic cell

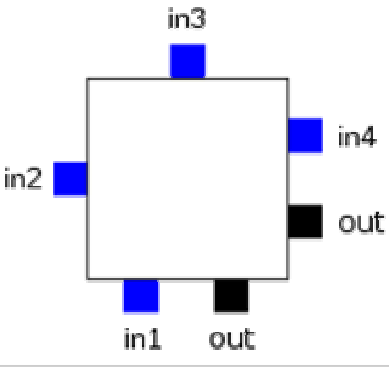
ALMs and Slices usually contains 2 or 4 structures similar to the example figure, with some shared signals.

CLBs/LABs typically contains a few ALMs/LEs/Slices.

In recent years, manufacturers have started moving to 6-input LUTs in their high performance parts, claiming increased performance.[30]

Since clock signals (and often other high-fanout signals) are normally routed via special-purpose dedicated routing networks in commercial FPGAs, they and other signals are separately managed.

For this example architecture, the locations of the FPGA logic block pins are shown below.



Logic Block Pin Locations

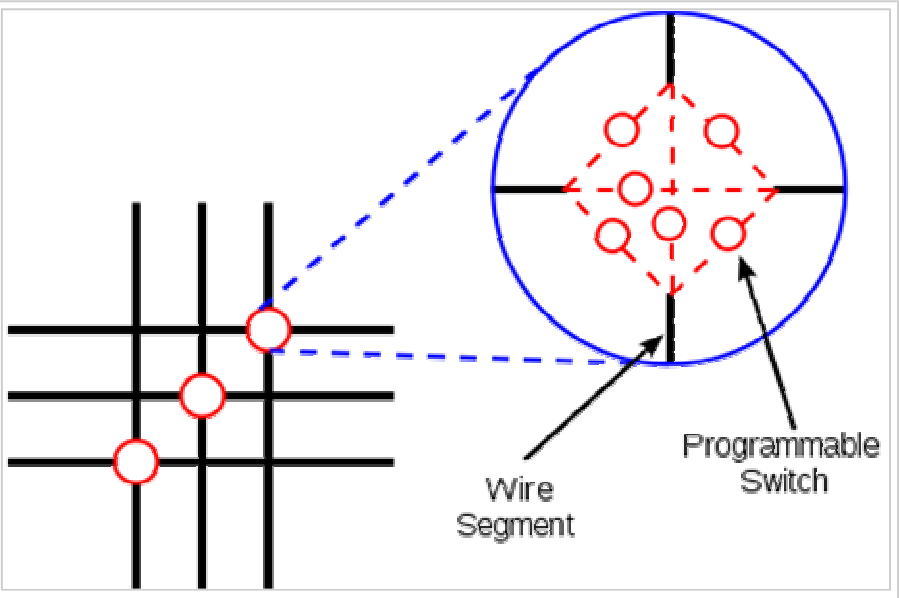
Each input is accessible from one side of the logic block, while the output pin can connect to routing wires in both the channel to the right and the channel below the logic block.

Each logic block output pin can connect to any of the wiring segments in the channels adjacent to it.

Similarly, an I/O pad can connect to any one of the wiring segments in the channel adjacent to it. For example, an I/O pad at the top of the chip can connect to any of the W wires (where W is the channel width) in the horizontal channel immediately below it.

Generally, the FPGA routing is unsegmented. That is, each wiring segment spans only one logic block before it terminates in a switch box. By turning on some of the programmable switches within a switch box, longer paths can be constructed. For higher speed interconnect, some FPGA architectures use longer routing lines that span multiple logic blocks.

Whenever a vertical and a horizontal channel intersect, there is a switch box. In this architecture, when a wire enters a switch box, there are three programmable switches that allow it to connect to three other wires in adjacent channel segments. The pattern, or topology, of switches used in this architecture is the planar or domain-based switch box topology. In this switch box topology, a wire in track number one connects only to wires in track number one in adjacent channel segments, wires in track number 2 connect only to other wires in track number 2 and so on. The figure below illustrates the connections in a switch box.



Switch box topology

Modern FPGA families expand upon the above capabilities to include higher level functionality fixed into the silicon. Having these common functions embedded into the silicon reduces the area required and gives those functions increased speed compared to building them from primitives. Examples of these include multipliers, generic DSP blocks, embedded processors, high speed IO logic and embedded memories.

FPGAs are also widely used for systems validation including pre-silicon validation, post-silicon validation, and firmware development. This allows chip companies to validate their design before the chip is produced in the factory, reducing the time-to-market.

FPGA design and programming

To define the behavior of the FPGA, the user provides a hardware description language (HDL) or a schematic design. The HDL form is more suited to work with large structures because it's possible to just specify them numerically rather than having to draw every piece by hand. However, schematic entry can allow for easier visualisation of a design.

Then, using an electronic design automation tool, a technology-mapped netlist is generated. The netlist can then be fitted to the actual FPGA architecture using a process called place-and-route, usually performed by the FPGA company's proprietary place-and-route software. The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated (also using the FPGA company's proprietary software) is used to (re) configure the FPGA.

Going from schematic/HDL source files to actual configuration: The source files are fed to a software suite from the FPGA/CPLD vendor that through different steps will produce a file. This file is then transferred to the FPGA/CPLD via a serial interface (JTAG) or to an external memory device like an EEPROM.

The most common HDLs are VHDL and Verilog, although in an attempt to reduce the complexity of designing in HDLs, which have been compared to the equivalent of assembly languages, there are moves to raise the abstraction level through the introduction of alternative languages. National Instrument's LabVIEW graphical programming language (sometimes referred to as "G") has an FPGA add-in module available to target and program FPGA hardware. The LabVIEW approach drastically simplifies the FPGA programming process [*citation needed*].

To simplify the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are commonly called *IP cores*, and are available from FPGA vendors and third-party IP suppliers (rarely free, and typically released under proprietary licenses). Other predefined circuits are available from developer communities such as OpenCores (typically released under free and open source licenses such as the GPL, BSD or similar license), and other sources.

In a typical design flow, an FPGA application developer will simulate the design at multiple stages throughout the design process. Initially the RTL description in VHDL or Verilog is simulated by creating test benches to simulate the system and observe results. Then, after the synthesis engine has mapped the design to a netlist, the netlist is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors. Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back-annotated onto the netlist.

Basic process technology types

- SRAM - based on static memory technology. In-system programmable and re-programmable. Requires external boot devices. CMOS.
- Antifuse - One-time programmable. CMOS.
- PROM - Programmable Read-Only Memory technology. One-time programmable because of plastic packaging.
- EPROM - Erasable Programmable Read-Only Memory technology. One-time programmable but with window, can be erased with ultraviolet (UV) light. CMOS.

- EEPROM - Electrically Erasable Programmable Read-Only Memory technology. Can be erased, even in plastic packages. Some but not all EEPROM devices can be in-system programmed. CMOS.
- Flash - Flash-erase EPROM technology. Can be erased, even in plastic packages. Some but not all flash devices can be in-system programmed. Usually, a flash cell is smaller than an equivalent EEPROM cell and is therefore less expensive to manufacture. CMOS.
- Fuse - One-time programmable. Bipolar.

Major manufacturers

Xilinx and Altera are the current FPGA market leaders and long-time industry rivals. Together, they control over 80 percent of the market,[31] with Xilinx alone representing over 50 percent.

Both Xilinx and Altera provide free Windows and Linux design software.[32][33]

Other competitors include Lattice Semiconductor (SRAM based with integrated configuration Flash, instant-on, low power, live reconfiguration), Actel (antifuse, flash-based, mixed-signal), SiliconBlue Technologies (extremely low power SRAM-based FPGAs with option integrated nonvolatile configuration memory), Achronix (http://www.achronix.com/) (RAM based, 1.5 GHz fabric speed) who will be building their chips on Intels' state-of-the art 22nm process[34], and QuickLogic (handheld focused CSSP, no general purpose FPGAs).

In March 2010, Tabula (http://www.tabula.com/) announced their new FPGA technology that uses time-multiplexed logic and interconnect for greater potential cost savings for high-density applications.

See also

- Gate array
- PSoC
- Application-specific integrated circuit (ASIC)
- Application-specific instruction-set processor (ASIP)
- Impulse CoDeveloper (Impulse C)
- Combinational logic
- Complex programmable logic device (CPLD)
- Software Defined Silicon (SDS)
- Soft processor
- FPGA prototype
- VHDL: VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
- Verilog: Hardware Description Language
- JHDL: Just-Another Hardware Description Language
- Reconfigurable Computing
- Partial re-configuration
- Hybrid-core computing
- Configware

- MyHDL Python based HDL—generates Verilog or VHDL
- SystemC System Description Language—C like
- Handel-C Extended C based description language designed for FPGAs
- Minimig open source re-implementation of an Amiga 500 using an FPGA
- Computing with Memory A time-multiplexed reconfigurable architecture using 2-D memory array
- Digital Clock Manager DCM - Digital Clock Management

References

1. ^ Wiśniewski, Remigiusz (2009). *Synthesis of compositional microprogram control units for programmable devices*. Zielona Góra: University of Zielona Góra. pp. 153. ISBN 978-83-7481-293-1.
2. ^ [a] [b] edu/~vaughn/challenge/fpga_arch.html FPGA Architecture for the Challenge]

3. ^ FPGA Signal Integrity tutorial (http://wiki.altium.com/display/ADOH/FPGA+SI+Tutorial+-+Simulating+the+Reflection+Characteristics)
4. ^ NASA: FPGA drive strength (http://klabs.org/richcontent/fpga_content/DesignNotes/signal_quality/actel_drive_strength/index.htm)
5. ^ Mike Thompson. "Mixed-signal FPGAs provide GREEN POWER" (http://www.eetimes.com/showArticle.jhtml?articleID=200000777) . EE Times, 2007-07-02.
6. ^ [a] [b] [c] [d] History of FPGAs (http://filebox.vt.edu/users/tmagin/history.htm)
7. ^ Google Patent Search, "Re-programmable PLA (http://www.google.com/patents?id=BB4vAAAAEBAJ&dq=4508977) ". Retrieved February 5, 2009.
8. ^ Google Patent Search, "Dynamic data re-programmable PLA (http://www.google.com/patents?id=1-gzAAAAEBAJ&dq=4524430) ". Retrieved February 5, 2009.
9. ^ Peter Clarke, EE Times, "Xilinx, ASIC Vendors Talk Licensing (http://www.eetimes.com/story/OEG20010622S0091) ." June 22, 2001. Retrieved February 10, 2009.
10. ^ [a] [b] [c] [d] [e] [f] Funding Universe. “Xilinx, Inc. (http://www.fundinguniverse.com/company-histories/Xilinx-Inc-Company-History.html) ” Retrieved January 15, 2009.
11. ^ Clive Maxfield, Programmable Logic DesignLine, "Xilinx unveil revolutionary 65nm FPGA architecture: the Virtex-5 family (http://www.pldesignline.com/products/187203173) . May 15, 2006. Retrieved February 5, 2009.
12. ^ Press Release, "Xilinx Co-Founder Ross Freeman Honored as 2009 National Inventors Hall of Fame Inductee for Invention of FPGA (http://press.xilinx.com/phoenix.zhtml?c=212763&p=irol-newsArticle&ID=1255523&highlight) "
13. ^ [a] [b] Clive Maxfield, book, "The Design Warrior's Guide to FPGAs (http://books.google.com/books?id=dnuwr2xOFpUC&pg=PA4&lpg=PA4&dq=FPGA+Market+growth+1990s&source=web&ots=YjFedB35Vp&sig= ".Published by Elsevier, 2004. ISBN 0750676043, 9780750676045. Retrieved February 5, 2009
14. ^ "Original Paper on Work, Thompson" (http://www.informatics.sussex.ac.uk/users/adrianth/ices96/paper.ps) . http://www.informatics.sussex.ac.uk/users/adrianth/ices96/paper.ps.
15. ^ [a] [b] Dylan McGrath, EE Times, "FPGA Market to Pass $2.7 Billion by '10, In-Stat Says (http://www.eetimes.com/news/design/business/showArticle.jhtml?articleID=188102617) ". May 24, 2006. Retrieved February 5, 2009.
16. ^ Narinder Lall eASIC Corporation, "FPGA Judgment Day:Rise of Second Generation Structured ASICs (http://www.opensystems-publishing.com/e-letter/dsp/2008/03/easic.pdf) . March, 2008. Retrieved February 5, 2009.
17. ^ Dylan McGrath, EE Times, "Gartner Dataquest Analyst Gives ASIC, FPGA Markets Clean Bill of Health (http://www.eetimes.com/conf/dac/showArticle.jhtml?articleID=164302400) ". June 13, 2005. Retrieved February 5, 2009.
18. ^ Virtex-4 Family Overview (http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf)
19. ^ Kuon, I.; Rose, J. (2006). *Measuring the gap between FPGAs and ASICs*. pp. 21. doi:10.1145/1117201.1117205 (http://dx.doi.org/10.1145%2F1117201.1117205) .
20. ^ [a] [b] Tim Erjavec, White Paper, "Introducing the Xilinx Targeted Design Platform: Fulfilling the Programmable Imperative (http://www.xilinx.com/publications/prod_mktg/Targeted_Design_Platforms.pdf) ." February 2, 2009. Retrieved February 2, 2009
21. ^ Huffmire Paper "Managing Security in FPGA-Based Embedded Systems (http://www2.computer.org/portal/web/csdl/doi/10.1109/MDT.2008.166) ." Nov-Dec 2008. Retrieved Sept 22, 2009
22. ^ "Virtex-5 FPGA Configuration User Guide" (http://www.xilinx.com/support/documentation/user_guides/ug191.pdf) . Xilinx Inc.. August 2010. pp. 33–35. http://www.xilinx.com/support/documentation/user_guides/ug191.pdf. Retrieved 2010-10-31.
23. ^ "Protecting Intellectual Property Through FPGA Design Security" (http://www.altera.com/literature/ads/fpgadesignsecurity.pdf) . Altera Corporation. http://www.altera.com/literature/ads/fpgadesignsecurity.pdf. Retrieved 2010-11-01.
24. ^ FPGA/DSP Blend Tackles Telecom Apps (http://www.bdti.com/articles/info_eet0207fpga.htm)
25. ^ Xilinx aims 65-nm FPGAs at DSP applications (http://www.eetimes.com/showArticle.jhtml?articleID=197001881)
26. ^ http://www.eecg.toronto.edu/~vaughn/challenge/fpga_arch.html
27. ^ http://www.altera.com/literature/hb/cyc2/cyc2_cii51002.pdf
28. ^ http://www.altera.com/literature/hb/stratix-iv/stx4_5v1_01.pdf
29. ^ http://www.xilinx.com/support/documentation/user_guides/ug070.pdf
30. ^ http://www.xilinx.com/bvdocs/whitepapers/wp245.pdf
31. ^ Seeking Alpha, "Altera and Xilinx Report: The Battle Continues (http://seekingalpha.com/article/85478-altera-and-xilinx-report-the-battle-continues) ". July 17, 2008. Retrieved February 5, 2009.
32. ^ "Xilinx ISE WebPACK" (http://www.xilinx.com/ise/logic_design_prod/webpack.htm) . http://www.xilinx.com/ise/logic_design_prod/webpack.htm.
33. ^ "Quartus II Web edition software" (https://www.altera.com/support/software/download/altera_design/quartus_we/dnl-quartus_we.jsp) . https://www.altera.com/support/software/download/altera_design/quartus_we/dnl-quartus_we.jsp.
34. ^ "Achronix to use Intel's 22 nm manufacturing" (http://newsroom.intel.com/community/intel_newsroom/blog/2010/11/01/chip-shot-achronix-to-use-

intel-s-22nm-manufacturing) . http://newsroom.intel.com/community/intel_newsroom/blog/2010/11/01/chip-shot-achronix-to-use-intel-s-22nm-manufacturing.

External links

- University of North Carolina at Charlotte's Reconfigurable Computing Laboratory (http://www.rcs.uncc.edu/wiki)
- FPGA Database (http://so-logic.net/en/knowledgebase/fpga_universe)